The best way to follow along this article is by downloading ADExplorer.exe from sysinternals and authenticate to your Active Directory.

Directory Servers

A directory server (more technically referred to as a Directory Server Agent, a Directory System Agent, or a DSA) is a type of network database that stores information represented as trees of entries. This is different from a relational database, which uses tables comprised of rows and columns, so directory servers may be considered a type of NoSQL database (even though directory servers have been around a lot longer than the term NoSQL has).

While virtually all directory servers support LDAP, some servers offer support for additional protocols that can be used to interact with the data. Some of these protocols include X.500 (the original Directory Access Protocol, for which LDAP is a much more lightweight version), naming service protocols like DNS and NIS, HTTP-based protocols like DSML and SCIM, and proprietary protocols like Novell's NDS.

The Directory Server interesting to us is of course Microsoft Active Directory.

Entries

An LDAP entry is a collection of information about an entity (like a user or a computer). Each entry consists of three primary components:

- a distinguished name (DN)
- a collection of attributes
- a collection of object classes.

DNs and RDNs

An entry's distinguished name, often referred to as a DN, uniquely identifies that entry and its position in the directory information tree (DIT) hierarchy. The DN of an LDAP entry is much like the path to a file on a filesystem.

An LDAP DN is comprised of zero or more elements called relative distinguished names, or RDNs. Each RDN is comprised of one or more (usually just one) attribute-value pairs. For example, "uid=john.doe" represents an RDN comprised of an attribute named "uid" with a value of "john.doe". If an RDN has multiple attribute-value pairs, they are separated by plus signs, like "givenName=John+sn=Doe".

The special distinguished name comprised of zero RDNs (and therefore has a string representation that is just an empty string) is sometimes called the "null DN" and references a special type of entry called the root DSE which provides information about the content and capabilities of the directory server.

For DNs with multiple RDNs, the order of the RDNs specifies the position of the associated entry in the DIT. RDNs are separated by commas, and each RDN in a DN represents a level in the hierarchy in descending order (i.e., moving closer to the root of the tree, which is called the naming context). That is, if you remove an RDN from a DN, you get the DN of the entry considered the parent of the former DN. For example, the DN "uid=john.doe,ou=People,dc=example,dc=com" has four RDNs, with the parent DN being "ou=People,dc=example,dc=com".

A Distinguished Name (DN) should be read from left to right. The RDNs more to the left represent the higher branches of a tree. The RDN to the far right represent the trunk of the tree.

The Distinguished Name of an active directory : `cn=Philip L,cn=users,dc=evilcorp,dc=local`
This is a Relative Distinguished Name: `dc=local`

Attributes

Attributes hold the data for an entry. Each attribute has an attribute type, zero or more attribute options, and a set of values that comprise the actual data.

Attribute types are schema elements that specify how attributes should be treated by LDAP clients and servers. All attribute types must have an object identifier (OID) and zero or more names that can be used to reference attributes of that type. They must also have an attribute syntax, which specifies the type of data that can be stored in attributes of that type, and a set of matching rules, which indicate how comparisons should be performed against values of attributes of that type. Attribute types may also indicate whether an attribute is allowed to have multiple values in the same entry, and whether the attribute is intended for holding user data (a user attribute) or is used for the operation of the server (an operational attribute). Operational attributes are typically used for configuration and/or state information.

Attribute options are not used all that often, but may be used to provide some metadata about an attribute. For example, attribute options may be used to provide different versions of a value in different languages.

Examples of attribute types are: `OctetString`, `Integer8`, `Integer`, `DirectoryString`, `Boolean`, `DN`, `OID`, `Sid`, `GeneralizedTime`.

Object Classes

Object classes are schema elements that specify collections of attribute types that may be related to a particular type of object, process, or other entity. Every entry has a structural object class, which indicates what kind of object an entry represents (e.g., whether it is information about a person, a group, a device, a service, etc.), and may also have zero or more auxiliary object classes that suggest additional characteristics for that entry.

Like attribute types, object classes must have an object identifier, but they may also have zero or more names. Object classes may also list a set of required attribute types (so that any entry with that object class must also include those attributes) and/or a set of optional attribute types (so that any entry with that object class may optionally include those attributes).

Object Identifiers (OIDs)

An object identifier (OID) is a string that is used to uniquely identify various elements in the LDAP protocol, as well as in other areas throughout computing. OIDs consist of a sequence of numbers separated by periods (e.g., "1.2.840.113556.1.4.473" is the OID that represents the server-side sort request control). In LDAP, OIDs are used to identify things like schema elements (like attribute types, object classes, syntaxes, matching rules, etc.), controls, and extended requests and responses. In the case of schema elements, there may also be user-friendly names that can be used in place of OIDs.

When you look at the Active Directory with ADExplorer it will automatically translate the OIDs to more userfriendly-names, so we won't have to lookup each OID to understand what it refers to.

Search Filters

Search filters are used to define criteria for identifying entries that contain certain kinds of information. There are a number of different types of search filters:

- Presence filters may be used to identify entries in which a specified attribute has at least one value.
- Equality filters may be used to identify entries in which a specified attribute has a particular value.
- Substring filters may be used to identify entries in which a specified attribute has at least one value that matches a given substring.
- Greater-or-equal filters may be used to identify entries in which a specified attribute has at least one value that is considered greater than or equal to a given value.
- Less-or-equal filters may be used to identify entries in which a specified attribute has at least one value that is considered less than or equal to a given value.
- Approximate match filters may be used to identify entries in which a specified attribute has a value that is approximately equal to a given value. Note that there is no official definition of “approximately equal to”, and therefore this behavior may vary from one server to another. Some servers use a “sounds like” algorithm like one of the Soundex or Metaphone variants.
- Extensible match filters may be used to provide more advanced types of matching, including the use of custom matching rules and/or matching attributes within an entry’s DN.
- AND filters may be used to identify entries that match all of the filters encapsulated inside the AND.
- OR filters may be used to identify entries that match at least one of the filters encapsulated inside the OR.
- NOT filters may be used to negate the result of the encapsulated filter (i.e., if a filter matches an entry, then a NOT filter encapsulating that matching filter will not match the entry, and if a filter does not match an entry, then a NOT filter encapsulating that non-matching filter will match the entry).

The logic used to perform the matching is encapsulated in matching rules, which are specified in attribute type definitions. Different matching rules may use different logic for making the determination. For example, the caseIgnoreMatch matching rule will ignore differences in capitalization when comparing two strings, while the caseExactMatch matching rule will not. Many matching rules are specific to certain data types (e.g., the distinguishedNameMatch matching rule expects to operate only on values that are DNs and can do things like ignore insignificant spaces between DN and RDN components, ignore differences in the order of elements in a multivalued RDN, etc.).

Search Base DNs and Scopes

All search requests include a base DN element, which specifies the portion of the DIT in which to look for matching entries, and a scope, which specifies how much of that subtree should be considered. The defined search scopes include:

- The baseObject scope (often referred to as just “base”) indicates that only the entry specified by the search base DN should be considered.
- The singleLevel scope (often referred to as “one” or “onelevel”) indicates that only entries immediately below the search base DN (but not the base entry itself) should be considered.
- The wholeSubtree scope (often referred to as “sub”) indicates that the entry specified as the search base DN and all entries below it (to any depth) should be considered.
- The subordinateSubtree scope indicates that all entries below the search base DN (to any depth), but not the search base entry itself, should be considered.

Controls

A control is a piece of information that can be included in an LDAP request or response to provide additional information about that request or response, or to change the way that it should be interpreted by the server (in the case of a request) or client (in the case of a response). For example, the server-side sort request control can be included in a search request to indicate that the server should sort the matching entries in a particular way before sending them to the client.

LDAP controls have three elements:

- An object identifier (OID) that uniquely identifies the type of control. This is a required element.
- A criticality. This is a flag that indicates how the server should behave if it does not recognize a provided request control, or if it cannot support the control in the context in which it was requested. A criticality of “true” indicates that the control is a critical part of the request, and that the server should reject the request if it cannot support the control. A criticality of “false” indicates that the control is more a “nice to have” part of the request, and that if the server cannot support the control then it should go ahead and process the operation as if the control had not been included. The criticality does not come into play if the server does support the control within the context of the request.
- An optional value, which can provide additional information for use in processing the control. For example, for a server-side sort request control, the control value should specify the desired sort order. The encoding for a control varies based on the type of control.

LDAPSEARCH Syntax

In order to authenticate to the LDAP server we need a username and password. The username is entered in the form of a Dinstingushed Name (DN) since that is a unique identifier. In the LDAP world this is called "bind dn". For example: `-D "cn=Philip L,cn=users,dc=evilcorp,dc=local"` .

The Search Base is specified with the `-b` flag in ldapsearch. For example: `-b "dc=evilcorp,dc=local"` .

ldapsearch will automatically default to the scope subTree, and the filter `objectClass=*` .

This is the basic syntax of a ldapsearch query: `usage: ldapsearch [options] [filter [attributes...]]` . So first the options, then the filter.

```
ldapsearch -h evilcorp.local  -w SecretPassword  -b "dc=evilcorp,dc=local" -D "cn=Philip L,cn=users,dc=evilcorp,dc=local" "(userPass
```

In order to be able to include fun operators like this:

```
(Operator(filter)(filter)(filter)...)
```

With `Operator` being `&` (AND), `|` (OR) and `!` (NOT).

```
ldapsearch -h evilcorp.local  -w SecretPassword  -b "dc=evilcorp,dc=local" -D "cn=Philip L,cn=users,dc=evilcorp,dc=local" "(|(userPa
```

```
ldapsearch -h <ip-address of dc> -w SecretPassword -b "dc=evilcorp,dc=local" -D "cn=Philip L,cn=users,dc=evilcorp,dc=local"
```